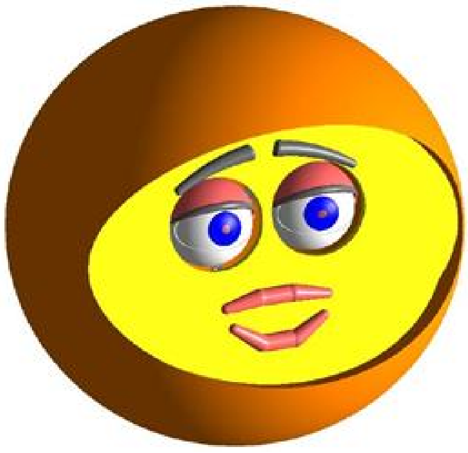
Speech interaction for an User Interface Robot

An interactive command & control based dialogue using MDP concept



M. Bani Aman

MSc Computer science

7 May 2002

Institute of Science
Intelligent Autonomous Systems (IAS) group
University of Amsterdam (UVA)
and
Heat, Mechanics and Particle optics group
Philips research (NatLab) Eindhoven

Speech interaction for an User Interface Robot

An interactive command & control based dialogue using MDP concept

Keywords **Speech recognition, speech synthesis, social context reasoning, dialogue management**

Supervisors **Frans Groen (UVA) and Ko Crucq (NatLab)**

M. Bani Aman

MSc Computer science

7 May 2002

Institute of Science
Intelligent Autonomous Systems (IAS) group
University of Amsterdam (UVA)
and
Heat, Mechanics and Particle optics group
Philips research (NatLab) Eindhoven

I dedicate my thesis

to

my mother

Abstract

This thesis discusses the design, implementation and evaluation of an interactive command & control dialogue system for an user interface robot functioning as a mediator between the users and all kind of appliances and services in a home environment. The speech module of this robot must recognize speech as input, do some reasoning about it and must synthesize speech as output to the user. Therefore, 3 modules are used to implement the speech module namely recognition, reasoning and synthesis module. The best configuration for this system is the use of L&H (Lernout& Hauspie) engine with the MDP (Markov Decision Process) concept. For the single-speaker-headset setup, L&H engine performs far better than the Microsoft engine gaining 12.5% more recognition rate up to 94.1% and using the same setup MDP has shown to be effective to improve the recognition performance by another 2.9% reaching rates of 97%, a total improvement of 15.4%. For the multiple-speaker-headset setup using MDP with L&H engine a recognition rate of 90.2% is achieved. Unfortunately no measurements could be applied to calculate the total improvement for this setup. Telex directional microphone seems to be not good enough to reach recognition rates above 80%. This minimum rate is required for a distance of at least 1 meter preventing the user from annoying repetition of the commands. Experiments show a drop of performance to values between 60% and 70% for distances between 50 and 100 cm. Therefore, another microphone setup must be applied.

Acknowledgements

First of all, I will thank my supervisor Frans Groen at UVA for his guidance and help during my project at Philips Research in Eindhoven. Also my gratitude to my supervisor Ko Crucq at Philips for his support and guidance for this project. Also my gratitude to members of IAS group for their remarks.

I will also thank my mother for her unprecedented support and help during my life, which has resulted to this work. Without her support this work never could be achieved.

Also my gratitude to my dear love Maria for her outstanding support and motivation during my project resulting to this work.

Contents

List of figures

List of tables

1. Introduction

1.1 Problem definition

This thesis, which describes my final project at Philips Research in Eindhoven, discuss the design, implementation and evaluation of an interactive command & control dialogue system for an user interface robot functioning as a mediator between the users and all kind of appliances and services in a home environment. In this first chapter the problem definition of the project and the state of the art technologies in this area will be exposed.

My final project was part of a much bigger project running at Philips research studying the use of a personal assistant in a home environment, as stated above. The personal assistant must be a human-like robot with skills similar to Aibo and Papero, which are made respectively by SONY and NEC. For instance, Papero can tell you the date, switch on the TV and show emotions such as happiness. The robot has to have vision skills to recognize faces, motion skills to follow a route and speech skills to start a conversion with the user. The latter one has been the goal of my final project i.e. giving the robot the speech skills to speak and listen to the user.

For the time being the robot's name will be 'Lino' to make referencing more convenient.

These speech skills must be implemented by a speech module that later will be integrated in a network of modules that creates the overall robot behavior. To show these speech skills different scenarios are used: Basics, Movement and Device Control. 'Basics' scenario contains commands such as "tell me the date" and "lino wake up", 'Movement' scenario contains commands such as "what is your position" and "lino go to the living room" and 'Device Control' scenario contains commands such as "switch on TV" and "set volume TV loud".
The speech module of Lino must recognize speech as input, do some reasoning about it and must synthesize speech as output to the user. Thus, speech module of Lino must consist of 3 modules namely recognition, reasoning and synthesis. These modules are depicted in figure 1.

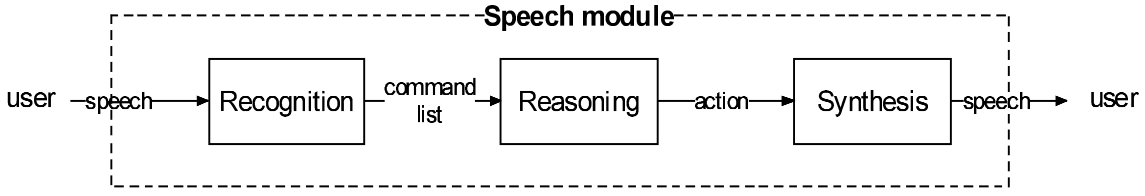


Figure 1: Speech module of Lino.

The recognition module must recognize speech and generate a list of commands with the highest matching scores. The reasoning module gets this list of commands as input and decide whether one of the commands in the list must be accepted or not as the spoken command. The output of this module is the corresponding action, which will be send as input to the synthesis module for speech generation. This action must contain the response of Lino as text that must be converted to speech by the synthesis engine. Another part of the action, which is not part of this thesis, are the control commands for controlling the household devices and mechanical commands for emotion generation or body movement.

Lino should have context awareness. Therefore, it should keep track of its internal state, which is updated by the action that is mentioned earlier, to put the commands into their context. For instance, when Lino is in the kitchen issuing the command ”switch on light” will be sufficient instead of the full command “switch on light in the kitchen”. This approach is more natural and convenient for the user.

A microphone setup should be used that can achieve good recognition results at a distance of 1 meter or more since this is the natural dialogue distance between people. Since the recognition rate also depend on the recognition engine, a study must be performed to find the best recognition engine available at this moment. As people prefer interactions that sounds natural, the choice for a good synthesis engine is also very important. Companies such as L&H (Lernout and Hauspie) and AT&T developed many good quality speech engines, so a study is necessary to find the best options for Lino. Since the speech module has to be developed using Microsoft SAPI platform, it is necessary to use engines that are compliant with this platform. Also the syntax of the grammar that implements the vocabulary, consisting of the commands that can be recognized by Lino, must be based on a standard that is compliant with this platform.
Also it must be noted that a way must be found for Lino to use power in a efficient way, such as bringing Lino in a sleeping state when it is not used, since the system is running on batteries.

The speech module will be integrated in a PVM (Parallel Virtual Machine) architecture, which uses a message-passing paradigm for communication between the modules. This network of modules uses the TCP/IP protocol and the connections are created by using sockets. A more detailed description about this architecture is out of the scope of this thesis.

1.2 Overview of the thesis

The next chapters are structured as follows. The second chapter gives a detailed description of concepts and methods that are used and why they are used for Lino. The third chapter describes how these concepts and methods are implemented and evaluated. The fourth chapter gives a conclusion about the results whether the used microphone and concept have been effective to solve the distance problem. The fifth chapter describes future work for possible extension of the dialogue management system. The sixth chapter contains references to the articles, which this thesis is based on and finally in the seventh chapter an appendix is included containing the vocabulary and the transition probabilities.

The reasoning module will be the focus of this thesis as it was the focus in my final project.

2. *Dialogue Management Architecture*

2.1 Overview

The Dialogue Management Architecture (DMA) integrates the speech architecture and the Dialogue Management (DM) concepts. The speech architecture consists of 3 modules namely Recognition, Reasoning and Synthesis module. These modules have already been introduced in the first chapter. In figure 2 a complete speech architecture of Lino is depicted, which contains the above modules and the interaction between them.

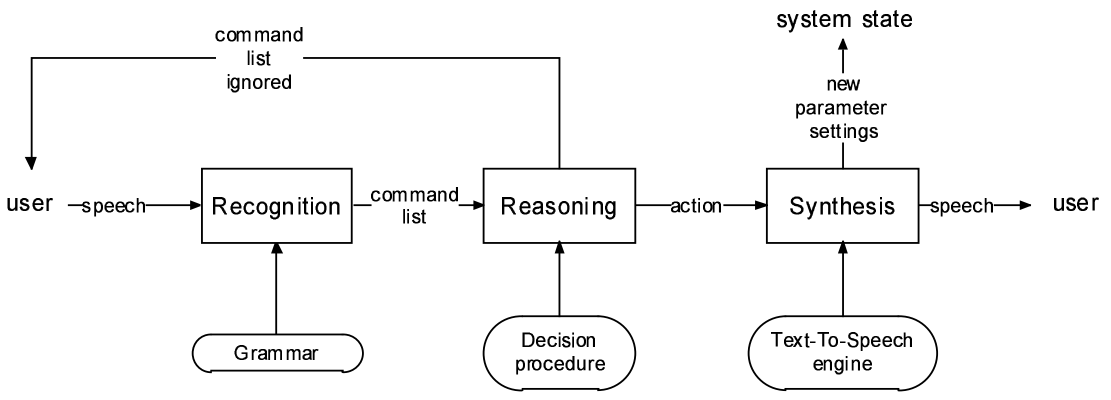


Figure 2: Speech architecture diagram of Lino.

Beside these 3 modules, 3 other components play an important role in the speech architecture. These components are Grammar, Decision procedure and Text-To-Speech engine. The Grammar component contains a context free grammar, which implements the vocabulary. The Decision procedure component describes the concept that is used to make the decisions and finally the Text-To-Speech (TTS) component realizes the actions by converting text to speech.
A dialogue session starts when the user has issued a command. Subsequently the recognition engine will search the vocabulary by using the grammar for the best word matches. Then a list of the best matching commands is created. This list will be the input for the reasoning module. In this module every command will be weighted and finally the best one will be accepted or will be ignored. After the acceptance of the best matching command the corresponding action will be channeled to the synthesis module. In this module the internal parameters will be set and texts will be converted into speech using a TTS engine.
To clarify this architecture a more detailed description of the modules and the components will be exposed later in this chapter.

As depicted in figure 2, synthesis module also updates internal parameters. These parameters keep track of Lino's internal state and position. Lino has 4 major states namely Sleeping , Idle, Listening and Semi listening state.

The state diagram can be depicted as follow:

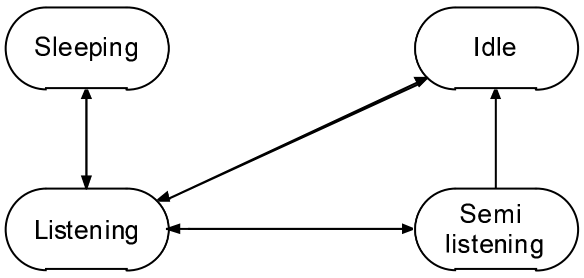


Figure 3: State diagram.

At start, Lino is in Sleeping state. Issuing command “Lino wake up” brings Lino in the Listening state. From the Listening state you can bring Lino in Idle state by issuing command “go down”. Semi listening state can be reached from the Listening state when after 10 seconds no commands are issued by the user. Subsequently the Semi listening state will reach the Idle state after another 10 seconds of silence. This construction is used to save power since Lino is running on batteries. Lino can always be brought back to Listening state from the Idle state by calling “Lino” and by issuing any command from the Semi listening state. Also one can send Lino to Sleeping state by calling “go sleep”. The difference between Sleeping and Idle state is in the duration of idleness. If Lino is not needed for a long period of time, lets say hours (no value is set for this), it can be better send to the Sleeping state for saving power otherwise Idle state is a more proper choice. No arrow is depicted between Idle and Sleeping state in the above figure since no time interval is defined. These major states can be described in more details as follow:

States	***Description***
Sleeping	In this state Lino is in deactivated mode and only a few of processes are active. In this state Lino can only react on the command “Lino wake up”.
Idle	In this state Lino is in standby mode and more processes are active than the sleeping state. In this state Lino can only react on the command “Lino”.
Listening	In this state Lino can deliver its services.
Semi listening	This state can be reached from the Listening state after a silence of 10 seconds. After an additional 10 seconds of silence the system will be transferred to Idle state.

Table 1: Major system states description.

In the Semi listening state the term silence refers to not accepting any commands. The time interval, which is used here namely the 10 seconds interval, is also known as attention span. In this attention span Lino is listening actively, which means that all the services that Lino deliver can be applied. After an additional 10 seconds of silence the system goes over to Idle state for saving power. The choice for 10 seconds intervals and the above states are based on the architecture of Bello [7]. The idea behind the choice of 10 seconds is that the attention span must be long enough to issue a command but must be short enough to save power. Saving power is necessary because Lino is running on batteries and inefficient use of power not only results in frequent recharge but also to a shorter life time for the batteries.

The Listening state is further divided into 10 sub states: basic, TV, radio, CD player, DVD player, video recorder, volume, open-close, position and light. These states are connected by a fully connected graph, so the states and the interactions do not need to be depicted but in appendix B a table will be given containing the transition probabilities between these states. In this table also transition probabilities between these sub states and Idle and Sleeping state are given. The following table describes sub states in more details:

Scenario	***Sub state***	***Description***
Basics	Basic	Contains the basic commands such as “tell me the time”.
Basics	Open-close	Contains commands to open and close objects such as doors.
Movement	Position	Contains commands to support movement of Lino.
Device Control	TV	Contains commands to control the TV.
Device Control	Radio	Contains commands to control the radio.
Device Control	CD player	Contains commands to control the CD player.
Device Control	DVD player	Contains commands to control the DVD player.
Device Control	Video recorder	Contains commands to control the video recorder.
Device Control	Volume	Contains commands to control the volume of the devices.
Device Control	Light	Contains commands to control the light in different places such as the kitchen.

Table 2: System sub states description with their corresponding scenario.

Sub states and their corresponding set of commands can be found in appendix A. The major states Sleeping, Idle and Semi listening are not part of the scenarios because the dialogue skills of Lino are only defined in these sub states.

In the next section the DM concepts, which are applied to Lino, will be discussed and an explanation will be given why these concepts are used for Lino. This is also the case for the speech modules later in this chapter.

2.2 Dialogue Management (DM) concepts

2.2.1 DM features

In a dialogue session the following DM features can be identified:

- *Overall-structure:*
  Generic structure; opening, body, closing.
- *Mixed initiative:*
  Turn of control between the user and the system, goal: clarifying.
- *Contextual interpretation:*
  Interpreting sentences according to context.
- *Error recovery:*
  Providing the system to recover from error.

Overall-structure You can 'open' the dialogue by waking up Lino. The 'body' of the dialogue is created by issuing commands and receiving the response. By letting Lino go to sleep the dialogue session can be 'closed'. This structure can also be generic when Lino is send to the Idle state and brought back several times to deliver services before sending him to sleep. Due to this structuring the dialogue sessions can be easier to analyze when the sessions become more complicated.
Mixed initiative makes it possible that not only the user can ask Lino to come with a response but also Lino can ask the user for additive information. This is for instance the case when Lino doesn't know what its position is and therefore it has to ask the user for feedback.
Contextual interpretation is necessary for Lino to put user utterances in their context. For instance when the user wants Lino to switch on the light in the kitchen and Lino is already in the kitchen then he doesn't need to clarify the position in the command, instead he wants Lino to keep track of its position. This approach is more natural and convenient for the user.
Error recovery is a very important feature because the robustness of the system depends on it. The errors that may occur in Lino must be handled in a proper way preventing it of crashes. For doing error recovery we have first to define errors that can appear in Lino's dialogue system. There are 2 different type of errors that may occur. First, a command can be accepted by Lino, which is not issued by the user. This can be an insertion error or a out-of-context error. An insertion error appears when a non-vocabulary command is mistaken by a vocabulary command and an out-

of-context error appears when a vocabulary command is mistaken by another vocabulary command. Second, the command that is issued by the user may not have been accepted. In Lino out-of-context errors are minimized by using the MDP concept, a detailed description of this concept will be exposed in section 2.4, and insertion errors are minimized by using non-vocabulary commands, but still no guarantee can be given that these errors will not occur. For the second type of error, recovery is on the user side i.e. the user can repeat the command a number of times till the command has been accepted. It is not the intention of Lino to create irritation for the user by let him to repeat the same command before it comes up with a reaction, but irritation is even more when Lino has to ask the user for repetition frequently. Therefore, experiments are performed and changes have been made to find the most optimal configuration for the system.

2.2.2 Difficulties modeling a dialogue

When modeling a dialogue some difficulties may arise:

- *Turn-taking:*
  Who must take the initiative in the dialogue?
- *Conversional fillers:*
  Conversational fillers are utterances such as ‘aha’, ‘yep’, ‘ok’ etc…. How should we handle these utterances?

In the current set of scenarios the initiative is always at the user side when a dialogue session starts. The initiative will only turn to Lino when Lino does not know its position. So turn-taking is well structured and will not cause problems.
By extending the grammar of Lino with junk-tags such as ‘junkBegin’ and ‘junkEnd’ the conversational fillers can be filtered and therefore a more natural conversation can be applied.

2.2.3 Discourse structure

As mentioned earlier, a dialogue session can be structured as follow:

Opening -> Body -> Closing

Further refinement can be done on the ‘Body’ part. This is also known as discourse structure and is given below:

Lesson -> Transaction -> Exchanges -> Moves -> Speech acts

Speech act can be a question. Move is a one or more speech acts such as question and answer pairs. Exchanges consists of one or more moves. Transactions consists of one or more Exchanges and Lessons consists of one or more Transactions.
This structure can be explained by taking as example a classroom with pupils and a teacher. A speech act occurs when the teacher says something to a pupil or the other

way around. A move occurs when the teacher says something to a pupil and the pupil says something back. Exchange is discussion about a topic with a pupil. A transaction is the discussion between the teacher and the pupil concerning more than one topic. A lesson can be seen as the discussion between the teacher and all the pupils in the class. If we take the teacher as Lino and the pupils as the users then this discourse structure can be applied to the dialogue sessions of Lino. This structure refinement is necessary to describe the whole dialogue session in more details. This structure is used for analyzes purposes only.

2.2.4 System prompt design

Two types of system prompts are available:

- Implicit: open-ended leaving the choice up to the user.
- Explicit: limiting the domain by indicating available options.

Implicit prompt suggest true knowledge of the system, while explicit prompt suggest lack of knowledge, therefore the available options must some how be exposed to the user.
To make this clear let's look at the next example. When you call your bank to find out the amount of money that you have on your account, the computer will list you the options that you can choose by selecting the appropriate number (explicit prompt). But after extensive use of the bank computer you have a good knowledge of the options, so exposing you with the options is waste of time. In this case you can just address the appropriate options without waiting too long, making the dialogue more convenient.
In case of Lino only the implicit prompt is used i.e. the user must know the vocabulary by heart to make optimal use of the services that Lino delivers. An interesting topic for future work could be the availability of explicit prompts, which make it possible that a user without any knowledge of the system can interact with Lino.

2.2.5 DM approach

Lino's DM make use of 'discourse grammar' approach i.e. the dialogue follows grammar rules. This limitation is restrained by the type of dialogue that is chosen namely Command & Control, which is discussed in the next section.

2.3 Speech Recognition Module

2.3.1 Recognition issues

Some issues that must be considered:

1. Should we use a speaker independent or a speaker dependent Speech Recognition Engine (SRE) ?

One of the advantages of a speaker independent SRE is that it can be used without the need for additional training by the user. This requires extensive training beforehand, where often a trade-off is made between the size of the target group and the recognition performance for this group. The performance will drop with the number of people it is trained for.
A speaker dependent SRE has the advantage that the recognizer performance usually is better than the performance of a speaker independent SRE. But this is only true for the people that trained the system, which obviously implies rather big recognition performance differences between different people.
Rypkema, Dixon and Kaufholz [1] indicate that people do not perceive training as a big problem. Of course, no training would be more convenient, but as long as the system needs to be trained only once, most people are willing to do so. All in all it can be concluded that training is an additional effort and is better avoided, unless the advantage for the user is clearly present.

For Lino a speaker independent approach is chosen to reach as many users. In the future a semi-independent approach can be used i.e. at start a speaker independent SRE is applied, where after the system is trained by the frequent users. This approach performs better than the speaker independent SRE and will perform less than the speaker dependent SRE but the advantage is in less training, which is more pleasant for the user.
An average recognition performance that now can be achieved with a speaker independent SRE is around 85% based on experiments described in many articles such as the article of Lamel [8].

2. Which type of recognition must be used: command & control or natural language?

A command & control speech recognizer is only able to recognize a specific set of commands, whereas a natural language speech recognizer is able to give meaning to naturally spoken text. The use of command language with an acceptable recognition performance is technically more feasible than natural language recognition. Rypkema, Dixon and Kaufholz [1] claim that instead of really talking to the system people want to be in control and therefore, for now, want to command their systems. The use of simple commands can be enforced using so-called power commands, which represent spoken macros or short cuts.

As Command & Control approach is technically more feasible and the way people want to interact with the robots is command-based, this approach will be used for Lino. The commands are defined in a vocabulary, which is given in the table in appendix A.

3. How must the vocabulary be defined to optimize the SRE performance?

The size of the vocabulary and the commands influence the performance of the SRE. When the size of a vocabulary increases the chance that 2 commands can sounds similar will increase too, which may result in wrong recognition. The commands of the vocabulary should not be too small to sound similar or too large to make recognition difficult. Therefore, a trade-off in size for both the vocabulary and the commands must be made.
For Lino a vocabulary of 148 commands is used. Although this vocabulary is not that big, word similarity problems can still arise. Therefore, medium to large commands are used to prevent this problem. The size of a command depends on the performance of the SRE. The better the performance, the larger a command can be.
Also the choice of proper commands is important. In case of Lino, some of the commands are adapted during the development of the dialogue system to improve the recognition. For instance, the command "switch on TV" is changed to "switch TV on" to make the chance lower to be mistaken with the command "switch off TV".
In Lino, at SRE level and thus not at user level, 'branching' is used i.e. the commands are divided into sub parts or keywords to increase the recognition performance. For example, "switch on TV" is decomposed in 2 parts. First part is "switch on" and the second part is "TV". This approach make the size of the vocabulary locally small due to the fact that the second part namely "TV" is just a device and only 7 devices are available, which means that the user utterance will be compared with only 7 commands instead of the whole vocabulary.
Another method that also can influence the performance of the SRE is the definition of the non-vocabulary commands also known as garbage collection. The user utterances that match garbage commands can be ignored. This method reduces the number of insertion errors, which will result in a better recognition performance. The system will also make insertion errors, when the vocabulary is too small.

4. How should we address the robot?

The choice for the name Lino was based on the Rypkema's survey [1]. According to Rypkema the name must be easy to remember and the pronunciation shouldn't be difficult. Also according to Diederiks in making of Bello [7] the name should not be too short to be confused with similar words and it should not be too common in everyday language.
Lino is short but not that common, so the chance is low to be mistaken with similar words. It is easy to remember and the pronunciation is simple and therefore a good choice for the robot.

2.3.2 The grammar

The (context free) grammar contains the commands from the vocabulary. Each grammar rule consist of a command and a reference to the corresponding action that must be taken when this command is recognized. The structure of a grammar rule is as follow:

Grammar rule = Command "action_reference"

Since branching is used for Lino more grammar rules are needed to describe the branched commands. The following grammar rules describe how a branched command is defined in the grammar:

Let's assume that we have a command 'com' that is described as:
com = part1 part2 part3

Then the grammar rules look like:

```
<com>        = part1  <com_part2>
<com_part2>  = part2  <com_part3>
<com_part3>  = part3  "action_reference"
```

Example of legal commands (square brackets indicate the branching):

[set volume] [TV] [loud]	"set volume TV loud"
[open] [the door]	"open the door"
[go] [down]	"go down"

The vocabulary can be found in appendix A.

2.3.3 How does recognition works?

First, the SRE must be initialized before starting with recognition. During the initialization the grammar must be parsed. Every textual command will be analyzed and corresponding feature vectors will be generated. The feature vectors can contain 10 to 20 parameters. The features (parameters) used are known as Mel-Frequency Cepstral Coefficients (MFCCs). These coefficients are obtained by taking the inverse Fourier transform of the log spectrum after it is warped according to the Mel scale. These feature vectors will be stored in a database that later can be used to match against the user utterances.

After the initialization the real recognition can begin. A speech waveform is generated when the user issues a command. This speech waveform will be sampled at rate between 6.6 kHz and 20 kHz and processed to generate a sequence of feature vectors, which are computed every 10 or 20 msec. These feature vectors will then be compared to feature vectors in the database to find the best word match.

2.4 Social Context Reasoning Module

The task of the Social Context Reasoning Module (SCRM) is speech understanding and performing the corresponding actions. As mentioned earlier, the description of basic speech skills was made by defining appropriate scenarios namely ‘Basics’, ‘Movement’ and ‘Device Control’.
Every scenario describes a set of commands. This set of commands is divided into subsets of commands, which are referred to as system states. This means that when one of the commands of the corresponding state is performed the system may change state from the current state to this state. These states are connected together by a fully connected graph. The transition from one state to another is described by a-priori probabilities. These probabilities are chosen manually, which are derived from the experiments. A better solution would be the use of neural networks for a better optimization.
For making the decision whether a command must be accepted or ignored is based on the Markov Decision Process (MDP) concept. Formally, MDP is described by:

- A finite set of states **S**,
- a finite set of actions **A**,
- a state transition function **T**: $\mathbf{S} \times \mathbf{A} \rightarrow \Pi(\mathbf{S})$,
  where $\Pi(\mathbf{S})$ is a probability distribution over **S** and $\mathbf{T}(s, a, s')$ is the probability of making a transition to state s' when taking action a in state s,
- a reward function R: $\mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow \mathbf{R}$,
  where $R(s, a, s')$ is the (expected) reward for making the transition $s \xrightarrow{a} s'$, and finally
- an optimal policy π^*: $\mathbf{S} \rightarrow \mathbf{A}$, which specifies the best action in each state. The optimal action in state s will be: $a^* = \arg\max_{a \in A} R(s, a, s')$.

The distinguishing characteristic of MDPs is the so called *Markov Property*, namely the property that state transitions are independent of any previous states and/or actions. Thus, optimal decisions can be made independently of previous decisions and/or transitions. In Lino this is also the case i.e. the next spoken command is not dependent of the previous one.

MDP is used as follow in Lino:

- A set of sub states **S** (user states),
- a finite set of commands **A** (members of the command list),
- a state transition function **T**: $\mathbf{S} \times \mathbf{A} \rightarrow \Pi(\mathbf{S})$,
  where $\Pi(\mathbf{S})$ is a probability distribution over **S** and $\mathbf{T}(s, a, s')$ is the probability of making a transition to state s' when taking action a in state s. The transition probability is a scalar value that is derived from the experiments manually, as stated above. These probabilities can be found in appendix B.
- a reward function R: $\mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow \mathbf{R}$,

where *R*(*s*, *a*, *s*') is the (expected) reward for making the transition $s \xrightarrow{a} s'$. The reward function is defined as *R*(*s*, *a*, *s*') = O(*k*, *b*) * T(*s*, *a*). Function O(*k*, *b*) stands for command observable normalized score that shows how good the match is with the input command and is generated by the recognition engine itself. The state *k* is the internal state of the engine itself and has nothing to do with the user state *s* and action *b* can be different from *a* since *b* is the input command to the engine and *a* is the estimated output command. The scores are generate in a domain between –100 and 100. To normalize these scores a normalization must be applied to map these scores to a domain between 0 and 1. The following formula describes the mapping:

$$W_i(k,b) = \frac{Score_i(k,b) + 100}{200}$$

Assuming that the commands not belonging to the list has a weight of 0 otherwise the mapping can not be realized.
The product of the command observable normalized score and the transition probability compute the (expected) reward, and finally

- an optimal policy π^*: **S** → **A**, which specifies the best action in each state. The optimal action in state *s* will be: $a^* = a$ where $\underset{\in}{\arg\max}$ *R*(*s*, *a*, *s*') > B. The scalar value B stands here for the boundary that must be compared against the highest reward. This boundary is also derived from the experiments manually and has a value equal to 0.028. The optimal action or command is the one with the highest reward that is larger than B. The optimal policy is calculated over a list of 5 commands, which is generated by the recognition engine.

The choice for this reward function using 2 parameters, namely 'O' and 'T', is based on the following arguments. When the observable normalized score is high but the transition probability is low the command should not be accepted because the recognized command is probably out of context error. But if the transition probability is high and the observable normalized score is low the command should not be accepted too because the recognized command maybe is some kind of background noise (low value for 'O'). So it is more convincing when both parameters have a high value i.e. the command is likely the one the user had issued and it fits the context or system state in which the user has been expected. This boundary can also be estimated by neural networks using labeled data sets i.e. when a command is accepted it will be marked as good data, otherwise the command will be marked as bad data.

2.5 Speech Synthesis Module

2.5.1 Overview

As mentioned earlier, the actions that must be taken given a command, can be found by using the action references that are part of the grammar rules. These references are linked during the initialization of the SRE. These actions generate not only speech but they also update the global variables, which maintain the context awareness. Speech generation is done by first defining the text response of Lino given some input command. Later, the text can be converted to speech using a TTS (Text-To-Speech) engine. In the next subsection a background in the theory of TTS is exposed.

2.5.2 Methods underlying TTS

In general 2 different methods are used:

1. *Diphone synthesis*
2. *Unit selection*

1. *Diphone synthesis*

This method uses diphones, which represents the transition from one phoneme to the next, to generate the speech. There are 2 steps involved in diphone synthesis, namely linguistic analysis and synthesis routines.
In the first step 4 actions will be taken: preprocessing, morphological analysis, grapheme-to-phoneme conversion and prosodic structure generation. *Preprocessing* organizes the input sentences into manageable lists of words, it also identifies numbers, abbreviations etc.. and transforms them into full text when needed. *Morphological analysis* propose all possible part-of-speech categories (noun, verb etc…) for each individual word based on their spelling. The number of possibilities is reduced by taking the context into account. Derived and compound words are decomposed into their elementary graphemic units (morphs) by simple regular grammars. *Grapheme-to-phoneme conversion* translates graphemes (letters) to their corresponding phonemes (sounds). This can be achieved by applying a set of rules to the input text. *Prosodic structure generation* specifies prosodic features of speech. Prosody refers to certain properties of the speech signal which are related to audible changes in pitch, loudness and syllable length.
In the second step 3 actions will be taken: concatenation, duration modification and pitch modification. *Concatenation* merges prerecorded small speech units, in this case diphones. *Duration modification* specifies the duration of all phonemes. *Pitch modification* specifies the melody of a sentence by changing the fundamental frequency pattern or pitch.
Drawback of diphone synthesis is lacking the naturalness of speech. This is caused by 3 factors. First of all, signal coding and manipulation after concatenation deteriorates the segmental quality due to spectral and phase mismatch. Second, audible discontinuities occur at concatenation points between diphones. Finally, the rules that

compute the pitch and duration targets often give a simplified view of prosody in natural speech.

2. *Unit selection*

This method involves the on-line selection of (variable-sized or non-uniform) acoustic units from a large speech database, which covers widespread phonetic sequences in various prosodic contexts. The goal of unit selection is to reduce the number of junctions and to reduce signal modifications.
Also unit selection has 2 steps. The steps are almost the same as diphone synthesis. Here the quality of the first step depends on the corpus, which should have a good phonetic and prosodic coverage of the intended domain, and the runtime selection criteria for the acoustic units. The corpus is usually many times larger than the traditional diphone databases, because it contains also sequences of diphones to reduce the number of concatenation which leads to a better speech quality. Selection of units works as follows: a synthesis specification is provided, giving a list of phonemes, prosodic features and values. The first stage involves, for each required (target) phoneme, looking at all phonemes with the same name that are in the database (candidate units). Each of this candidate phonemes is assigned a target cost, based on the similarity of the target. A second type of cost., concatenation cost, is calculated based on how well two units join together (lower cost equals a smoother join). A network can be constructed and costs as described above are assigned to each unit and to links between units. The task is then to find the lowest cost path through the network.

Comparison

Unit selection reduces the number of junctions and signal modifications which leads to a better speech quality that approximate the natural speech far better than when using diphone synthesis. Therefore, the SSE (Speech Synthesis Engine) that is used for Lino is based on unit selection.

3. Implementation and evaluation

3.1 Overview

In this chapter the implementation of the methods and the concepts, which have been discussed in the previous chapter, will be described and evaluated.
Let's consider figure 2. As we can see, for implementing the recognition module we have to choose a recognition engine and define a grammar. Also a proper microphone setup must be chosen. Subsequently reasoning module is implemented by using MDP and finally the synthesis engine is implemented by choosing a proper TTS engine. Therefore, these components will be discussed in the next section. Also the programming environment that integrates these modules will be discussed.

For evaluation 2 approaches are used: qualitative and quantitative approach. Qualitative approach relies on user's opinion of the system. For instance, in case of Lino, the user can be asked about the performance of the system? It must be noted that different people has different interpretation of the system.
Quantitative approach is divided to black box and glass box evaluation. Black box method is concerned with input/output behavior whereas the glass box method is concerned with behavior of individual components in the system. For Lino the black box method is used to evaluate the engines due to the absence of development packages and glass box method is used to evaluate the MDP concept.

For the evaluation of the system different tests are defined:

- Different user test (test 1)
- Headset distance test (test 2)
- Directional microphone distance test (test 3)
- Noise on the background test (test 4)
- Collective test (test5)
    - Microsoft SRE without MDP test
    - L&H SRE without MDP test
    - L&H SRE with MDP test

With these tests the performance of the system is measured in different environments and with different implementations. More detailed description about these tests can be found in section 3.3.

3.2 Implementation

Microphones

Since Lino must be a human-like robot, it needs to have ears to listen. Therefore, the use of directional microphones is more desirable than a headset. At the other hand, it is not convenient for the user to put on and off the headset when respectively he want to speak and to stop with speaking. The requirement for these directional microphones is achieving recognition rates above 80% for a distance of 1 meter, otherwise it can be very annoying for the user. A Telex USB M-560 directional microphone is used for this purpose. The choice for a directional and not a standard microphone is in the high performance that can be delivered. Unfortunately it is only sensitive in one direction, the user direction. This problem can be solved by a sound localizer. Lino can localize the sound with the sound localizer and turn to the direction of the sound. Since the sound localizer was not part of my final project it will not be discussed in this thesis. It must be noted that the sound localizer is part of the overall project and will later be integrated in Lino. The headset, which also is used, is an Andrea Solution recognition headset and it will be used only for comparison purposes.

Grammar

A context free grammar is used to implement the vocabulary. The syntax of the grammar is based on a standard, which is compliant with the Microsoft SAPI platform.

Speech recognition engine

The speech recognition engine, which is used in Lino, is Lernout & Hauspie (L&H). A comparison between this SRE and Microsoft SRE will be given in the evaluation section. These engines use Microsoft SAPI platform. This platform is used due to compatibility with a high number of speech engines, which in the future make replacement easier of the current engines with the better ones.

MDP implementation

The algorithm implementing MDP concept is as follow:

- Read the command list.
- Calculate for each action (command) the reward value.
- Choose the command with the highest reward.
- Compare this reward with the boundary B, if reward > B command is accepted otherwise it is ignored.

Programming environment

Visual basic is used to implement the code. Since this programming language is interface based, the implementation can easy be tested and visualized. In the future a C++ version is necessary to embed the program into the PVM platform, which is coded in C++ and therefore can not support visual basic programs.

Speech synthesis engine

For the time being the SSE of Microsoft is used. In the future this engine must be changed by an AT&T SSE. The choice of AT&T SSE is based on the evaluation, which is given in the evaluation section.

3.3 Evaluation

3.3.1 Overview

First of all, we have to define cases to evaluate the system. These cases must contain a sequence of commands that must be spoken by the user. Three cases are used with the first case as the most general one whereas the other 2 are more specific. The second case focuses on the controlling of the devices such as TV and radio and the third case focuses more on the controlling of the play devices such as CD player and video recorder.

The 3 cases are given below:

	Case 1	**Case 2**	**Case 3**
1	lino wake up	lino wake up	lino wake up
2	come here	tell me the date	tell me the time
3	tell me the date	dance for me	switch cd player on
4	tell me the time	switch alarm on	play cd
5	switch tv on	switch tv on	stop cd
6	set volume loud	set volume tv higher	pause cd
7	next channel	next channel	set track to 7
8	set channel to cnn	previous channel	set volume soft
9	set channel to mtv	set channel to net5	set track to 13
10	switch radio on	set channel to rtl4	set track to 24
11	set volume tv lower	set volume tv lower	set volume cd player higher
12	set volume radio higher	set volume tv off	next track
13	switch cd player on	switch radio on	set volume lower
14	set volume cd player soft	switch tv off	switch dvd player on
15	play cd	set volume normal	previous track
16	next track	set volume radio on	switch video recorder on
17	set track to 6	open the door	play dvd
18	set track to 14	open the frontdoor	play video
19	previous track	close the window	stop dvd
20	stop cd	close the door	pause video
21	switch dvd player on	what is your position	fast forward video
22	play dvd	you are in the bedroom	switch cd player off
23	stop dvd	lino go to the kitchen	rewind video

24	switch video recorder on	what is your position	switch radio on
25	play video	switch light on	stop dvd
26	pause video	switch light on in the livingroom	go away
27	stop video	close the frontdoor	switch dvd player off
28	rewind video	thank you	go sleep
29	fast forward video	switch alarm off	
30	give me the list of activated devices	go away	
31	open the door	go down	
32	close the window	lino	
33	what is your position	go away	
34	you are in the kitchen	go sleep	
35	what is your position		
36	lino go to the living room		
37	what is your position		
38	switch light on		
39	switch light on in the bedroom		
40	switch radio off		
41	switch cd player off		
42	go sleep		

Table 3: Evaluation cases.

For the evaluation also we have to define how the recognition rate is calculated. This calculation is done after performing each case. The recognition rate is calculated as follows:

T	=	Total number of errors
I	=	Insertion errors
O	=	Out-of-context errors
E	=	Error rate
R	=	Recognition rate
N	=	Total number of commands in a case

$$T = I + O$$

$$E = \frac{T}{N}$$

$$R = 100\% - E$$

In the next subsection the evaluation procedure for choosing the best SSE is given and later in this chapter the following tests are discussed:

- Different user test (test 1)
- Headset distance test (test 2)
- Directional microphone distance test (test 3)

- Noise on the background test (test 4)
- Collective test (test5)
    - Microsoft SRE without POMDP test
    - L&H SRE without POMDP test
    - L&H SRE with POMDP test

3.3.2 Choosing the appropriate SSE

To choose the appropriate engine we have to evaluate speech synthesis engines using the unit selection method. Searching on internet leads to 2 technology providers that are promising: AT&T and L&H.
Since we do not have the development packages of these engines (i.e. we cannot examine the components of the engines separately) we have to use black-box evaluation method. For evaluating the engines using black-box evaluation method the following aspects can be taken into account:

1. Are the words pronounced correctly? If not, then there has been a mistake in the grapheme-to-phoneme conversion module.

2. Does the emphasis fall on the correct syllable? If not, then there must be a mismatch in F0 (base frequency) or energy in unit selection.

3. Are the correct words accented? If not, then this may either be a mistake in the prediction of accents, or a gap in the speech database.

4. Are there any mismatches in F0? If this is the case, then the speech will sound not fluent and can seriously hamper the comprehension of the message.

Evaluating of both engines has been done in aspect order, which is given above. Many sentences are used to make the evaluation as good as possible. Results are given below:

Aspect 1: AT&T makes no mistake in pronouncing the words correctly, where L&H makes few mistakes.
Aspect 2: Both of them perform well for this aspect.
Aspect 3: Both of them perform well for this aspect.
Aspect 4: L&H sample speech sounds not fluent and some words are difficult to comprehend.

Conclusion:
AT&T has a better speech synthesizer than L&H, so in the future the TTS of AT&T should be used for Lino.

3.3.3 Different user test

This test is the most general one and thus will use the above 3 cases. Also different users were asked to perform these cases. It must be noted that this test is done by using L&H engine and MDP concept. The used microphone is the headset at a distance of 1 cm.
This test measure the performance of the system by using 4 phases. In the first phase the slow version and in the second phase the fast version is used. The slow version describes the situation when the next command should be issued after the previous one is completed. In the fast version the next command can be issued while the previous one is still executing. The choice for these 2 versions is based on the fact that some users may speak faster and some more slowly with Lino. The question would be then if the system performance remains the same. The third phase is an assignment case that consists of a sequence of tasks that must be performed without having the list of commands at hand. This is done to see how fast the users can learn the vocabulary. The fourth and the last phase consists of a questionnaire, in which the users will be asked to give their impression about the system.

Evaluation conditions

1. Commands must be issued in a sequential manner. During the first evaluation the next command can only be issued when the execution of the previous one already is completed. At the other hand, during the second evaluation one can proceed with the next command while the previous one is still executing. In this second evaluation it is forbidden to issue the next command immediately after the previous one. In this situation the recognizer thinks that the two subsequent commands belong together, which cause the recognition to fail.
2. The commands should be pronounced in a normal tempo, thus not too slowly or too fast.
3. The commands must be spoken as clear as possible.
4. Experiments will be done in a quiet room.

Assignment case

In this case the users will be asked to perform the tasks, which are given below. This is done to see how fast the users can learn the commands.

	Assignment Case
1	wake up lino
2	switch tv on
3	change the volume of the tv to loud
4	change channel to cnn
5	switch cd player on
6	change the volume of the cd player to higher
7	play the cd
8	go to next track

9	change track to 12
10	switch video recorder on
11	rewind the video
12	stop the cd
13	ask lino for time
14	ask lino for date
15	ask lino for its position
16	tell lino its position
17	ask lino to move to the livingroom
18	switch the light on in the livingroom
19	switch tv off
20	ask lino for activated devices
21	change track to 7
22	switch cd player off
23	let lino go to sleep

Table 4: Assignment case.

A questionnaire

A questionnaire is given below to see how the users think about the system performance. The users are asked to be honest in answering the questions, otherwise this evaluation will not have any value.

1. Do you need to repeat the commands frequently?
2. If so, did you find it annoying?
3. Do you think you need to adapt yourself too much to the system?
4. Did you find the system fast enough or was it annoying slow?
5. Do you prefer to issue the commands during the execution of the previous command or do you prefer to wait till the previous command is completed?
6. Does the system sounds natural?
7. Do you think you can learn the system fast to control it?
8. If not, do you think that there are too many commands and/or the commands are difficult to remember?
9. Do you feel yourself convenient when using the system?

Case evaluation result

The following figures describe the test results for slow, fast, average and new slow version of the implementation using L&H recognition engine with MDP concept. The average version is the average of the slow and the fast version. Finally, in the new slow version the syntax of some of the commands is changed to improve the performance of the system.

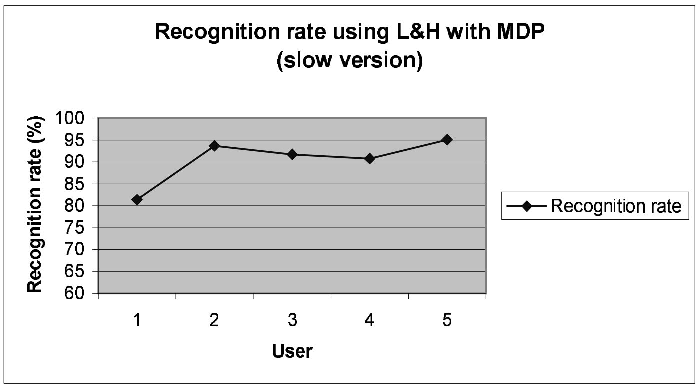


Figure 4: Recognition rate of the slow version using L&H recognition engine with MDP concept.

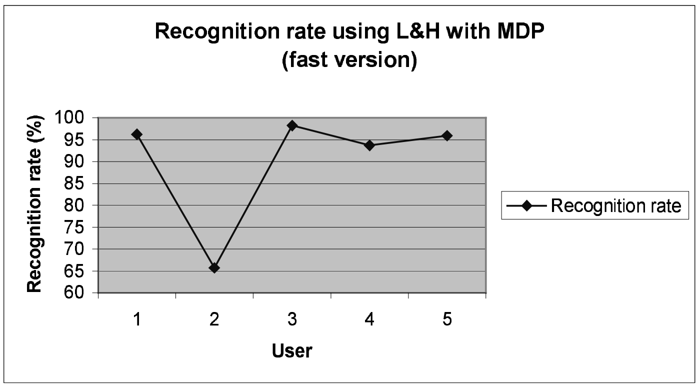


Figure 5: Recognition rate of the fast version using L&H recognition engine with MDP concept.

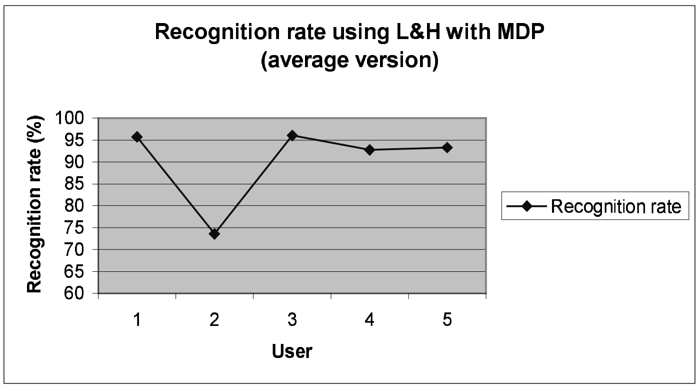


Figure 6: Recognition rate of the average version using L&H recognition engine with MDP concept.

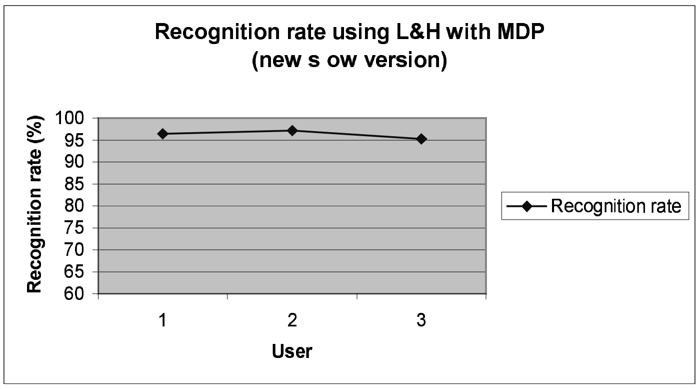


Figure 7: Recognition rate of the new slow version using L&H recognition engine with MDP concept. Unfortunately the other 2 users could not be present for this test.

Questionnaire evaluation result

The majority of the users did not found the system annoying due to frequent repetition of the commands. Only one user in the group of 5 was not happy with the system performance because the system could not handle the heavy English accent of this user. Also the majority of the users indicated that it was not necessary to adapt themselves too much to the system. They found the system fast, convenient to use and the vocabulary easy to learn. They found that the system didn't sound natural. Finally they prefer at the beginning to issue the commands one after each waiting for the previous command to complete to see the reaction of the system but in the later stage, if the system can handle it, they will speak faster not waiting for the previous command to complete. Why should we wait if we can go on with speaking?

Evaluation discussion for this test

As we can see from figure 4 the system has for slow version a recognition rate between 90%-95% for 4 of the 5 users and for one user it is 81%. This user is the same user with the heavy English accent, which was mentioned earlier, and will be referred as user H. On average a recognition rate of 90.5% is achieved.
For fast version, figure 5, the system perform worse for user H and better for the rest of the users. Probably when user H speak faster the accuracy of his pronunciation decrease drastically, which had leaded to drop of performance from 81% to 66%. Therefore on average for fast version the performance is dropped from 90.5 % to 89.9%.
Figure 6 shows the average of the above versions. An overall average on this give us a recognition rate of 90.2%.
In figure 7 the recognition rate of the system is given after a change on the syntax of the commands. Users 1, 2 and 3 in this figure are respectively the same as users 3, 2 and 4 in the previous figures. As we can see a better result is gained. On average an increase of 4.3% in recognition performance is gained.

3.3.4 Headset distance test

This test is done by using L&H recognition engine with MDP concept. The microphone that is used is the headset. The test is performed at headset distances 1, 3, 5, 8 and 10 cm. Only one user (in this case myself) will perform this test due to lack of time. Indeed a more accurate result can be gained with more users.

Evaluation conditions

1. Commands must be issued in a sequential manner i.e. the next command can only be issued when the execution of the previous one already is completed (slow version).
2. The commands should be pronounced in a normal tempo, thus not to slowly or to fast.
3. The commands must be spoken as clear as possible.
4. Experiments will be done in a quiet room.

Applied cases

In this test only slow versions of case 1 and 3 are used.

Case evaluation result

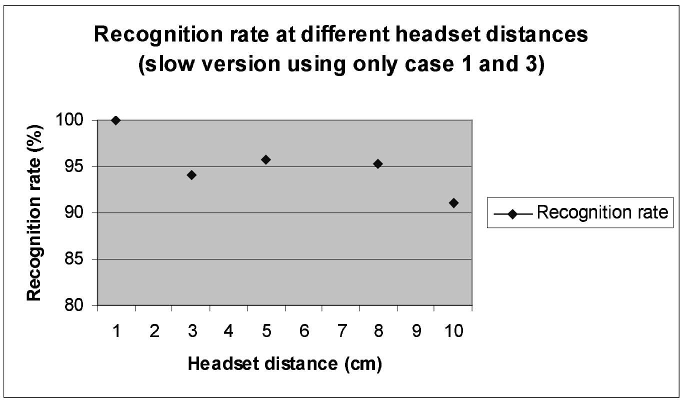


Figure 8: Recognition rate of the slow version using case 1 and 3 at different headset distances. Also here L&H engine with MDP concept is used. Only one user had performed this test.

Evaluation discussion for this test

As one can predict the recognition rate decrease when the headset distance increases. The recognition rate has been dropped for about 10% after the headset distance is changed from 1 to 10 cm.

3.3.5 Directional microphone distance test

This test is done by using L&H engine with MDP concept. The microphone that is used is the directional microphone. The test is performed at headset distances 10, 30, 50, 70 and 100 cm. Only one user (in this case myself) will perform this test due to lack of time. Indeed a more accurate result can be gained with more users.

Evaluation conditions

1. Commands must be issued in a sequential manner i.e. the next command can only be issued when the execution of the previous one already is completed (slow version).
2. The commands should be pronounced in a normal tempo, thus not to slowly or to fast.
3. The commands must be spoken as clear as possible.
4. Experiments will be done in a quiet room.

Applied cases

In this test only slow versions of case 1 and 3 are used.

Case evaluation result

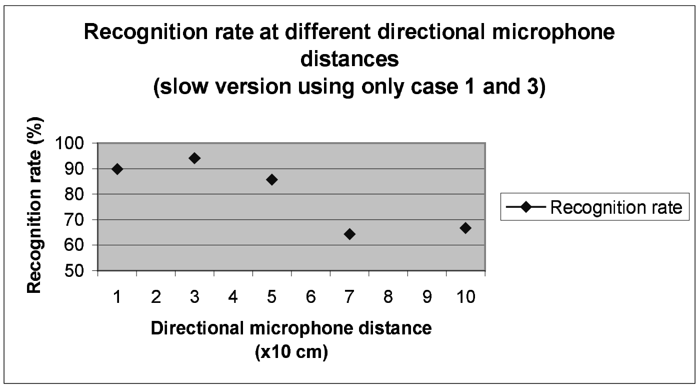


Figure 9: Recognition rate of the slow version using case 1 and 3 at different directional microphone distances. Also here L&H engine with MDP concept is used. Only one user had performed this test.

Evaluation discussion for this test

As we can see from figure 9 the recognition rate after 50 cm decreases drastically reaching rates between 60% and 70%. Since at least a recognition rate of 80% at a distance of 1 meter or more must be reached to prevent the system to be annoying for the user, this microphone can not be good enough to use. Therefore, another microphone setup should be applied.

3.3.6 Noise on the background test

This test is done by using L&H recognition engine with MDP concept. The microphone that is used is the headset. The test is performed at headset distance 1 cm. Only one user (in this case myself) will perform this test due to lack of time. Indeed a more accurate result can be gain with more users.

Evaluation conditions

1. Commands must be issued in a sequential manner i.e. the next command can only be issued when the execution of the previous one already is completed (slow version).
2. The commands should be pronounced in a normal tempo, thus not to slowly or to fast.
3. The commands must be spoken as clear as possible.
4. Experiments will be done in a quiet room with music on the background.

Applied cases

In this test only slow version of case 1 is used.

Case evaluation result

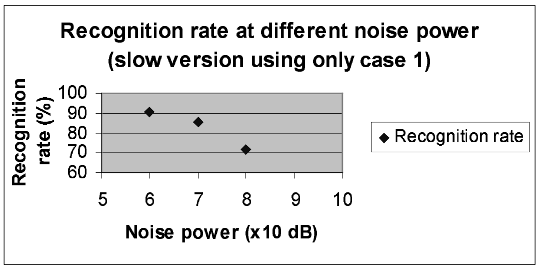


Figure 10: Recognition rate of the slow version using only case 1 with different dB values for background noise. Also here L&H engine with MDP concept is used. Only one user had performed this test.

Evaluation discussion for this test

As we can see from figure 10 the recognition performance is sensitive for background noise. The recognition rate is 70% at 80 dB, which means that when you want to speak with Lino and there is music on the background at normal volume level then the dialogue can be a bit annoying. Therefore, background noise must be limited while talking to Lino.

3.3.7 Collective test

This collective test consists of the following tests:

- Microsoft SRE without MDP test
- L&H SRE without MDP test
- L&H SRE with MDP test

The first test is performed by using Microsoft recognition engine without MDP concept. The second one is performed by using L&H recognition engine without MDP concept and finally the last one is performed by using L&H recognition engine with MDP concept. For all of these tests a headset as microphone is used. A headset distance of 1 cm is chosen. Only one user (in this case myself) has performed these tests due to lack of time. Indeed a more accurate result can be gained with more users.

Evaluation conditions

The evaluation conditions are the same for all 3 tests. The conditions are as follow:

1. Commands must be issued in a sequential manner i.e. the next command can only be issued when the execution of the previous one already is completed (slow version).
2. The commands should be pronounced in a normal tempo, thus not to slowly or to fast.
3. The commands must be spoken as clear as possible.
4. Experiments will be done in a quiet room.

Applied cases

For all these 3 tests only slow version of case 1 and 3 is used.

Case evaluation result

Test	***Case 1***	***Case 3***	***Average***
MS without MDP	81.0	82.1	81.6
L&H without MDP	95.2	92.9	94.1
L&H with MDP	97.6	96.4	97.0

Table 5: Recognition rates of slow versions using only case 1 and 3. Three tests are used namely MS without MDP, L&H without MDP and L&H with MDP.

Evaluation discussion for this test

As we can see in table 3 the average recognition rate that can be achieved with MS SRE without MDP are below the average rates (85%). Therefore, for Lino a better recognition engine, L&H SRE, is used to improve the performance.

The average recognition rate of the system with L&H SRE is drastically improved with 12.5% from 81.6% to 94.1%, which means that L&H SRE is far better than MS SRE. But the system can still be improved and therefore MDP concept is used.
As we can see the MDP concept has improved the system performance with 2.9% from 94.1% to 97.0%.

4. Conclusion

For the single-speaker-headset setup, L&H engine performs far better than the Microsoft engine gaining 12.5% more recognition rate up to 94.1% and using the same setup MDP has shown to be effective to improve the recognition performance by another 2.9% reaching rates of 97%, a total improvement of 15.4%. For the multiple-speaker-headset setup using MDP with L&H engine a recognition rate of 90.2% is achieved. Unfortunately no measurements could be applied to calculate the total improvement for this setup. Telex directional microphone seems to be not good enough to reach recognition rates above 80%. This minimum rate was required for a distance of at least 1 meter preventing the user from annoying repetition of the commands. Experiments show a drop of performance to values between 60% and 70% for distances between 50 and 100 cm. Therefore, another microphone setup must be applied. Also we have seen that the system with its most optimal configuration, using L&H SRE with MDP concept, is still sensitive for background noise. So, background noise must be avoided as much as possible.

5. Future work

The performance of the system can still be improved and therefore the following improvements are proposed.
As mentioned earlier, neural networks can be used for a better estimation of the transition probabilities and the reward boundary. The reward boundary could be estimated by using neural networks with labeled data sets. Also the use of another reward function may result in a better performance.
Further improvement can be gained by using better (directional) microphones, which can achieve a good recognition rate at a distance of at least 1 meter. To make the system to sound more natural the SSE of Microsoft can be replaced by the SSE of AT&T as it was concluded in the third chapter. Also the system can be trained for frequent users to improve the performance. Implicit system prompt design can be extended by explicit one omitting the need of knowing the entire vocabulary by heart.

6. References

[1] J. Rypkema, P. Dixon, P. Kaufholz, *Using automatic speech recognition in consumer products,* Nat.Lab. Report 7033, Eindhoven, June 1998.

[2] X. Wang, L. Pols, *A preliminary study about robust speech recognition for a robotics application*, Institute of Phonetic Sciences, University of Amsterdam, Proceedings 21 (1997), 11-20.

[3] N. Roy, J. Pineau, S. Thrun, *Spoken Dialog Management for robots*, Robotics Institute, Carnegie Mellon University. Proceedings of the ACL, October 2000.

[4] G. Churcher, E. Atwall, C. Souter, *Dialogue Management Systems: a Survey and Overview*, Report 97.06, University of Leeds, February 1997.

[5] A. Kellner, S. Martin, P. Philips, T. Portele, B. Souvignier, *Conversational User Interfaces: SPICE- a first research prototype*, PFL-Aachen Report 1463/2000, July 2000.

[6] R. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, V. Zue, *Survey of the State of the Art in Human Language Technology*. Center for Spoken Language Understanding CSLU, Carnegie Mellon University, Pittsburgh, PA, 1995.

[7] E. Diederiks, A. Montvay, R. van de Sluis, K. Vrielink, *Speech Recognition in WWICE: The Making of "Bello"*. Nat.Lab. Technical Note NL-TN 2000/236, Oktober 2000.

[8] L. Lamel, *Spoken language dialog system development and evaluation at LIMSI*, Proc Internat. Symposium on Spoken Dialogue, Sydney, Australia, November 1998.

[9] E. Klabbers, *State of the art in speech generation*, IPO: Center for User-System Interaction, Eindhoven University of Technology and Philips, February 2001.

[10] E. Krahmer, *The Science and Art of Voice Interfaces*, IPO: Center for User-System Interaction, Eindhoven University of Technology and Philips, year 2001.

[11] T. Bailey, M. Lagoudakis, N. Popoviciu, *Using Markov Decision Processes and Reinforcement Learning to Solve the "Remote Versus Local Execution" Problem*, Department of Computer Science, Duke University, Durham, USA.

7. Appendix

A. Vocabulary

*** = not defined

State	**Sub state**	**Command**
SLEEPING	***	go sleep
IDLE	***	go down
SEMI LISTENING	***	***
LISTENING	Basic	Lino
LISTENING	Basic	lino wake up
LISTENING	Basic	thank you
LISTENING	Basic	come here
LISTENING	Basic	go away
LISTENING	Basic	dance for me
LISTENING	Basic	tell me the date
LISTENING	Basic	tell me the time
LISTENING	Basic	give me the list of activated devices
LISTENING	Basic	switch alarm on
LISTENING	Basic	switch alarm off
LISTENING	TV	switch TV on
LISTENING	TV	switch TV off
LISTENING	TV	switch television on
LISTENING	TV	switch television off
LISTENING	TV	next channel
LISTENING	TV	previous channel
LISTENING	TV	set channel to NED1
LISTENING	TV	set channel to NED2
LISTENING	TV	set channel to NED3
LISTENING	TV	set channel to RTL4
LISTENING	TV	set channel to RTL5
LISTENING	TV	set channel to SBS6
LISTENING	TV	set channel to NET5
LISTENING	TV	set channel to FOX8
LISTENING	TV	set channel to YORIN
LISTENING	TV	set channel to CNN
LISTENING	TV	set channel to BBC1
LISTENING	TV	set channel to BBC2
LISTENING	TV	set channel to TMF
LISTENING	TV	set channel to MTV
LISTENING	TV	set volume TV on
LISTENING	TV	set volume TV off
LISTENING	TV	set volume television on

LISTENING	TV	set volume television off
LISTENING	TV	set volume TV soft
LISTENING	TV	set volume television soft
LISTENING	TV	set volume TV normal
LISTENING	TV	set volume television normal
LISTENING	TV	set volume TV loud
LISTENING	TV	set volume television loud
LISTENING	TV	set volume TV higher
LISTENING	TV	set volume television higher
LISTENING	TV	set volume TV lower
LISTENING	TV	set volume television lower
LISTENING	Radio	switch radio on
LISTENING	Radio	switch radio off
LISTENING	Radio	set volume radio on
LISTENING	Radio	set volume radio off
LISTENING	Radio	set volume radio soft
LISTENING	Radio	set volume radio normal
LISTENING	Radio	set volume radio loud
LISTENING	Radio	set volume radio higher
LISTENING	Radio	set volume radio lower
LISTENING	CD player	switch CD player on
LISTENING	CD player	switch CD player off
LISTENING	CD player	set volume CD player on
LISTENING	CD player	set volume CD player off
LISTENING	CD player	set volume CD player soft
LISTENING	CD player	set volume CD player normal
LISTENING	CD player	set volume CD player loud
LISTENING	CD player	set volume CD player higher
LISTENING	CD player	set volume CD player lower
LISTENING	CD player	play CD
LISTENING	CD player	stop CD
LISTENING	CD player	pause CD
LISTENING	CD player	next track
LISTENING	CD player	previous track
LISTENING	CD player	set track to 1
LISTENING	CD player	set track to 2
LISTENING	CD player	set track to 3
LISTENING	CD player	set track to 4
LISTENING	CD player	set track to 5
LISTENING	CD player	set track to 6
LISTENING	CD player	set track to 7
LISTENING	CD player	set track to 8
LISTENING	CD player	set track to 9
LISTENING	CD player	set track to 10
LISTENING	CD player	set track to 11

LISTENING	CD player	set track to 12
LISTENING	CD player	set track to 13
LISTENING	CD player	set track to 14
LISTENING	CD player	set track to 15
LISTENING	CD player	set track to 16
LISTENING	CD player	set track to 17
LISTENING	CD player	set track to 18
LISTENING	CD player	set track to 19
LISTENING	CD player	set track to 20
LISTENING	CD player	set track to 21
LISTENING	CD player	set track to 22
LISTENING	CD player	set track to 23
LISTENING	CD player	set track to 24
LISTENING	CD player	set track to 25
LISTENING	CD player	set track to 26
LISTENING	CD player	set track to 27
LISTENING	CD player	set track to 28
LISTENING	CD player	set track to 29
LISTENING	CD player	set track to 30
LISTENING	DVD player	switch DVD player on
LISTENING	DVD player	switch DVD player off
LISTENING	DVD player	set volume DVD player on
LISTENING	DVD player	set volume DVD player off
LISTENING	DVD player	set volume DVD player soft
LISTENING	DVD player	set volume DVD player normal
LISTENING	DVD player	set volume DVD player loud
LISTENING	DVD player	set volume DVD player higher
LISTENING	DVD player	set volume DVD player lower
LISTENING	DVD player	play DVD
LISTENING	DVD player	stop DVD
LISTENING	DVD player	pause DVD
LISTENING	Video recorder	switch video recorder on
LISTENING	Video recorder	switch video recorder off
LISTENING	Video recorder	record video
LISTENING	Video recorder	rewind video
LISTENING	Video recorder	fast forward video
LISTENING	Video recorder	play video
LISTENING	Video recorder	pause video
LISTENING	Video recorder	stop video
LISTENING	Volume	set volume on
LISTENING	Volume	set volume off
LISTENING	Volume	set volume soft
LISTENING	Volume	set volume normal
LISTENING	Volume	set volume loud
LISTENING	Volume	set volume higher

LISTENING	Volume	set volume lower
LISTENING	Open-Close	open the door
LISTENING	Open-Close	close the door
LISTENING	Open-Close	open the front door
LISTENING	Open-Close	close the front door
LISTENING	Open-Close	open the window
LISTENING	Open-Close	close the window
LISTENING	Position	what is your position
LISTENING	Position	you are in the living room
LISTENING	Position	you are in the kitchen
LISTENING	Position	you are in the bedroom
LISTENING	Position	lino go to the living room
LISTENING	Position	lino go to the bedroom
LISTENING	Position	lino go to the kitchen
LISTENING	Light	switch light on
LISTENING	Light	switch light off
LISTENING	Light	switch light on in the living room
LISTENING	Light	switch light off in the living room
LISTENING	Light	switch light on in the bedroom
LISTENING	Light	switch light off in the bedroom
LISTENING	Light	switch light on in the kitchen
LISTENING	Light	switch light off in the kitchen

B. State transition probabilities

Current state	Next state	Probability
Basic	Basic	0.094
Basic	TV	0.087
Basic	Radio	0.087
Basic	CD player	0.087
Basic	DVD player	0.087
Basic	Video recorder	0.087
Basic	Volume	0.058
Basic	Open-Close	0.087
Basic	Position	0.087
Basic	Light	0.087
Basic	Sleeping	0.072
Basic	Idle	0.080
TV	Basic	0.090
TV	TV	0.090
TV	Radio	0.090
TV	CD player	0.090
TV	DVD player	0.090
TV	Video recorder	0.090
TV	Volume	0.090
TV	Open-Close	0.075
TV	Position	0.075
TV	Light	0.075
TV	Sleeping	0.075
TV	Idle	0.075
Radio	Basic	0.097
Radio	TV	0.097
Radio	Radio	0.097
Radio	CD player	0.097
Radio	DVD player	0.097
Radio	Video recorder	0.097
Radio	Volume	0.097
Radio	Open-Close	0.081
Radio	Position	0.081
Radio	Light	0.081
Radio	Sleeping	0.081
Radio	Idle	0.081
CD player	Basic	0.090
CD player	TV	0.090
CD player	Radio	0.090
CD player	CD player	0.090
CD player	DVD player	0.090

CD player	Video recorder	0.090
CD player	Volume	0.090
CD player	Open-Close	0.075
CD player	Position	0.075
CD player	Light	0.075
CD player	Sleeping	0.075
CD player	Idle	0.075
DVD player	Basic	0.090
DVD player	TV	0.090
DVD player	Radio	0.090
DVD player	CD player	0.090
DVD player	DVD player	0.090
DVD player	Video recorder	0.090
DVD player	Volume	0.090
DVD player	Open-Close	0.075
DVD player	Position	0.075
DVD player	Light	0.075
DVD player	Sleeping	0.075
DVD player	Idle	0.075
Video recorder	Basic	0.094
Video recorder	TV	0.094
Video recorder	Radio	0.094
Video recorder	CD player	0.094
Video recorder	DVD player	0.094
Video recorder	Video recorder	0.102
Video recorder	Volume	0.031
Video recorder	Open-Close	0.079
Video recorder	Position	0.079
Video recorder	Light	0.079
Video recorder	Sleeping	0.079
Video recorder	Idle	0.079
Volume	Basic	0.081
Volume	TV	0.081
Volume	Radio	0.081
Volume	CD player	0.081
Volume	DVD player	0.081
Volume	Video recorder	0.081
Volume	Volume	0.113
Volume	Open-Close	0.081
Volume	Position	0.081
Volume	Light	0.081
Volume	Sleeping	0.081
Volume	Idle	0.081
Open-Close	Basic	0.095
Open-Close	TV	0.079

Open-Close	Radio	0.079
Open-Close	CD player	0.079
Open-Close	DVD player	0.079
Open-Close	Video recorder	0.079
Open-Close	Volume	0.079
Open-Close	Open-Close	0.111
Open-Close	Position	0.079
Open-Close	Light	0.079
Open-Close	Sleeping	0.079
Open-Close	Idle	0.079
Position	Basic	0.095
Position	TV	0.079
Position	Radio	0.079
Position	CD player	0.079
Position	DVD player	0.079
Position	Video recorder	0.079
Position	Volume	0.079
Position	Open-Close	0.079
Position	Position	0.111
Position	Light	0.079
Position	Sleeping	0.079
Position	Idle	0.079
Light	Basic	0.085
Light	TV	0.085
Light	Radio	0.085
Light	CD player	0.085
Light	DVD player	0.085
Light	Video recorder	0.085
Light	Volume	0.085
Light	Open-Close	0.085
Light	Position	0.085
Light	Light	0.099
Light	Sleeping	0.070
Light	Idle	0.070
Sleeping	Basic	1
Sleeping	TV	0
Sleeping	Radio	0
Sleeping	CD player	0
Sleeping	DVD player	0
Sleeping	Video recorder	0
Sleeping	Volume	0
Sleeping	Open-Close	0
Sleeping	Position	0
Sleeping	Light	0
Sleeping	Sleeping	0

Sleeping	Idle	0
Idle	Basic	1
Idle	TV	0
Idle	Radio	0
Idle	CD player	0
Idle	DVD player	0
Idle	Video recorder	0
Idle	Volume	0
Idle	Open-Close	0
Idle	Position	0
Idle	Light	0
Idle	Sleeping	0
Idle	Idle	0